# Deep Variational Reinforcement Learning for POMDPs

**Maximilian Igl** [1]  **Luisa Zintgraf** [1]  **Tuan Anh Le** [1]  **Frank Wood** [2]  **Shimon Whiteson** [1]
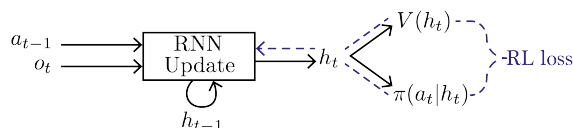
## Abstract

Many real-world sequential decision making problems are partially observable by nature, and the environment model is typically unknown. Consequently, there is great need for reinforcement learning methods that can tackle such problems given only a stream of rewards and incomplete and noisy observations. In this paper, we propose deep variational reinforcement learning (DVRL), which introduces an inductive bias that allows an agent to learn a generative model of the environment and perform inference in that model to effectively aggregate the available information. We develop an $n$-step approximation to the evidence lower bound (ELBO), allowing the model to be trained jointly with the policy. This ensures that the latent state representation is suitable for the control task. In experiments on Mountain Hike and flickering Atari we show that our method outperforms previous approaches relying on recurrent neural networks to encode the past.

**(a) RNN-based approach.** The RNN acts as an encoder for the action-observation history, on which actor and critic are conditioned. The networks are updated end-to-end with an RL loss.



**(b) DVRL.** The agent learns a generative model which is used to update a belief distribution. Actor and critic now condition on the belief. The generative model is learned to optimise both the ELBO *and* the RL loss.

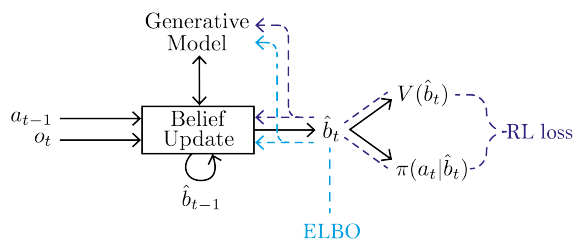**Figure 1:** Comparison of RNN and DVRL encoders.

## 1. Introduction

Most deep reinforcement learning (RL) methods assume that the state of the environment is fully observable at every time step. However, this assumption often does not hold in reality, as occlusions and noisy sensors may limit the agent's perceptual abilities. Such problems can be formalised as partially observable Markov decision processes (POMDPs) (Astrom, 1965; Kaelbling et al., 1998). Because we usually do not have access to the true generative model of our environment, there is a need for reinforcement learning methods that can tackle POMDPs when only a stream of observations is given, without any prior knowledge of the latent state space or the transition and observation functions.

POMDPs are notoriously hard to solve: since the current observation does in general not carry all relevant information for choosing an action, information must be aggregated over time and in general, the entire history must be taken into account.

This history can be encoded either by remembering features of the past (McCallum, 1993) or by performing inference to determine the distribution over possible latent states (Kaelbling et al., 1998). However, the computation of this *belief state* requires knowledge of the model.

Most previous work on deep learning in POMDPs relies on training a recurrent neural network (RNN) to summarise the past. Examples are the deep recurrent Q-network (DRQN) (Hausknecht & Stone, 2015) and the action-specific deep recurrent Q-network (ADRQN) (Zhu et al., 2017). Because these approaches are completely model-free, they place a heavy burden on the RNN. Since performing inference implicitly requires a known or learned model, they are likely to summarise the history either by only remembering features of the past or by computing simple heuristics instead of actual belief states. This is often suboptimal in complex tasks. Generalisation is also often easier over beliefs than

---
[1]University of Oxford, United Kingdom [2]University of British Columbia, Canada. Correspondence to: Maximilian Igl <maximilian.igl@eng.ox.ac.uk>.

over trajectories since distinct histories can lead to similar or identical beliefs.

The premise of this work is that deep policy learning for POMDPs can be improved by taking less of a black box approach than DRQN and ADRQN. While we do not want to assume prior knowledge of the transition and observation functions or the latent state representation, we want to allow the agent to learn models of them and infer the belief state using these learned models.

To this end, we propose DVRL, which implements this approach by providing a helpful inductive bias to the agent. In particular, we develop an algorithm that can learn an internal generative model and use it to perform approximate inference to update the belief state. Crucially, the generative model is not only learned based on an ELBO objective, but also by how well it enables maximisation of the expected return. This ensures that, unlike in an unsupervised application of variational autoencoders (VAEs), the latent state representation and the inference performed on it are suitable for the ultimate control task. Specifically, we develop an approximation to the ELBO based on autoencoding sequential Monte Carlo (AESMC) (Le et al., 2018), allowing joint optimisation with the $n$-step policy gradient update. Uncertainty in the belief state is captured by a particle ensemble. A high-level overview of our approach in comparison to previous RNN-based methods is shown in Figure 1.

We evaluate our approach on Mountain Hike and several *flickering* Atari games. On Mountain Hike (a low dimensional, continuous environment), we can show that DVRL is better than an RNN based approach at inferring the required information from past observations for optimal action selection in a simple setting. Our results on flickering Atari show that this advantage extends to complex environments with high dimensional observation spaces. Here, partial observability is introduced by (1) using only a single frame as input at each time step and (2) returning a blank screen instead of the true frame with probability 0.5.

## 2. Background

In this section, we formalise POMDPs and provide background on recent advances in VAEs that we use. Lastly, we describe the policy gradient loss based on $n$-step learning and A2C.

### 2.1. Partially Observable Markov Decision Processes

A partially observable Markov decision process (POMDP) is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, F, U, R, b_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, and $\mathcal{O}$ the observation space. We denote as $s_t \in \mathcal{S}$ the latent state at time $t$, and the distribution over initial states $s_0$ as $b_0$, the initial belief state. When an action $a_t \in \mathcal{A}$ is executed, the state changes accord-

ing to the transition distribution, $s_{t+1} \sim F(s_{t+1}|s_t, a_t)$. Subsequently, the agent receives a noisy or partially occluded observation $o_{t+1} \in \mathcal{O}$ according to the distribution $o_{t+1} \sim U(o_{t+1}|s_{t+1}, a_t)$, and a reward $r_{t+1} \in \mathbb{R}$ according to the distribution $r_{t+1} \sim R(r_{t+1}|s_{t+1}, a_t)$.

An agent acts according to its policy $\pi(a_t|o_{\leq t}, a_{<t})$ which returns the probability of taking action $a_t$ at time $t$, and where $o_{\leq t} = (o_1, \ldots, o_t)$ and $a_{<t} = (a_0, \ldots, a_{t-1})$ are the observation and action histories, respectively. The agent's goal is to learn a policy $\pi$ that maximises the expected future return

$$J = \mathbb{E}_{p(\tau)}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t\right], \tag{1}$$

over trajectories $\tau = (s_0, a_0, \ldots, a_{T-1}, s_T)$ induced by its policy[1], where $0 \leq \gamma < 1$ is the discount factor. We follow the convention of setting $a_0$ to *no-op* (Zhu et al., 2017).

In general, a POMDP agent must condition its actions on the entire history $(o_{\leq t}, a_{<t})$ which grows exponentially in $t$. This can be accomplished by memory based approaches, for example by using suffix trees (McCallum & Ballard, 1996; Shani et al., 2005; Bellemare et al., 2014; Bellemare, 2015; Messias & Whiteson, 2017). However, those approaches are only suitable for small discrete observation spaces and suffer from large memory requirements.

Alternatively, it is possible to infer the filtering distribution $p(s_t|o_{\leq t}, a_{<t}) =: b_t$, called the *belief state*. This is a sufficient statistic of the history that can be used as input to an optimal policy $\pi^\star(a_t|b_t)$. The belief space does not grow exponentially, but the belief update step requires knowledge of a model:

$$b_{t+1} = \frac{\int b_t U(o_{t+1}|s_{t+1}, a_t) F(s_{t+1}|s_t, a_t) \mathrm{d}s_t}{\int \int b_t U(o_{t+1}|s_{t+1}, a_t) F(s_{t+1}|s_t, a_t) \, \mathrm{d}s_t \, \mathrm{d}s_{t+1}}. \tag{2}$$

### 2.2. Variational Autoencoder

We define a family of priors $p_\theta(s)$ over some latent state $s$ and decoders $p_\theta(o|s)$ over observations $o$, both parameterised by $\theta$. A variational autoencoder (VAE) learns $\theta$ by maximising the sum of log marginal likelihood terms $\sum_{n=1}^{N} \log p_\theta(o^{(n)})$ for a dataset $(o^{(n)})_{n=1}^{N}$ where $p_\theta(o) = \int p_\theta(o|s)p_\theta(s) \, \mathrm{d}s$ (Rezende et al., 2014; Kingma & Welling, 2014)) . Since evaluating the log marginal likelihood is intractable, the VAE instead maximises a sum of ELBOs where each individual ELBO term is a lower bound on the log marginal likelihood,

$$\mathrm{ELBO}(\theta, \phi, o) = \mathbb{E}_{q_\phi(s|o)}\left[\log \frac{p_\theta(o|s)p_\theta(s)}{q_\phi(s|o)}\right], \tag{3}$$

---

[1]The trajectory length $T$ is stochastic and depends on the time at which the agent-environment interaction ends.

for a family of encoders $q_\phi(s|o)$ parameterised by $\phi$. This objective also forces $q_\phi(s|o)$ to approximate the posterior $p_\theta(s|o)$ under the learned model. Gradients of (3) are estimated by Monte Carlo sampling with the reparameterisation trick (Kingma & Welling, 2014; Rezende et al., 2014).

## 2.3. VAE for Time Series

For sequential data, we assume that a series of latent states $s_{\leq T}$ gives rise to a series of observations $o_{\leq T}$. We consider a family of generative models parameterised by $\theta$ that consists of the initial distribution $p_\theta(s_0)$, transition distribution $p_\theta(s_t|s_{t-1})$ and observation distribution $p_\theta(o_t|s_t)$. Given a family of encoder distributions $q_\phi(s_t|s_{t-1}, o_t)$, we can also estimate the gradient of the ELBO term in the same manner as in (3), noting that:

$$p_\theta(s_{\leq T}, o_{\leq T}) = p_\theta(s_0) \prod_{t=1}^{T} p_\theta(s_t|s_{t-1})p_\theta(o_t|s_t), \quad (4)$$

$$q_\phi(s_{\leq T}|o_{\leq T}) = p_\theta(s_0) \prod_{t=1}^{T} q_\phi(s_t|s_{t-1}, o_t), \quad (5)$$

where we slightly abuse notation for $q_\phi$ by ignoring the fact that we sample from the model $p_\theta(s_0)$ for $t = 0$. Le et al. (2018), Maddison et al. (2017) and Naesseth et al. (2018) introduce a new ELBO objective based on sequential Monte Carlo (SMC) (Doucet & Johansen, 2009) that allows faster learning in time series:

$$\text{ELBO}_{\text{SMC}}(\theta, \phi, o_{\leq T}) = \mathbb{E}\left[\sum_{t=1}^{T} \log\left(\frac{1}{K}\sum_{k=1}^{K} w_t^k\right)\right], \quad (6)$$

where $K$ is the number of particles and $w_t^k$ is the weight of particle $k$ at time $t$. Each particle is a tuple containing a weight $w_t^k$ and a value $s_t^k$ which is obtained as follows. Let $s_0^k$ be samples from $p_\theta(s_0)$ for $k = 1, \ldots, K$. For $t = 1, \ldots, T$, the weights $w_t^k$ are obtained by resampling the particle set $(s_{t-1}^k)_{k=1}^K$ proportionally to the previous weights and computing

$$w_t^k = \frac{p_\theta(s_t^k|s_{t-1}^{u_{t-1}^k})p_\theta(o_t|s_t^k)}{q_\phi(s_t^k|s_{t-1}^{u_{t-1}^k}, o_t)}, \quad (7)$$

where $s_t^k$ corresponds to a value sampled from $q_\phi(\cdot|s_{t-1}^{u_{t-1}^k}, o_t)$ and $s_{t-1}^{u_{t-1}^k}$ corresponds to the resampled particle with the ancestor index $u_0^k = k$ and $u_{t-1}^k \sim \text{Discrete}((w_{t-1}^k/\sum_{j=1}^{K} w_{t-1}^j)_{k=1}^K)$ for $t = 2, \ldots, T$.

## 2.4. A2C

One way to learn the parameters $\rho$ of an agent's policy $\pi_\rho(a_t|s_t)$ is to use $n$-step learning with A2C (Wu et al.,

2017), the synchronous simplification of asynchronous advantage actor-critic (A3C) (Mnih et al., 2016). An actor-critic approach can cope with continuous actions and avoids the need to draw state-action sequences from a replay buffer. The method proposed in this paper is however equally applicable to other deep RL algorithms.

For $n$-step learning, starting at time $t$, the current policy performs $n_s$ consecutive steps in $n_e$ parallel environments. The gradient update is based on this mini-batch of size $n_e \times n_s$. The target for the value-function $V_\eta(s_{t+i}), i = 0, \ldots, n_s - 1$, parameterised by $\eta$, is the appropriately discounted sum of on-policy rewards up until time $t + n_s$ and the off-policy bootstrapped value $V_\eta^-(s_{t+n_s})$. The minus sign denotes that no gradients are propagated through this value. Defining the advantage function as

$$A_\eta^{t,i}(s_{t+i}, a_{t+i}) := \left(\sum_{j=0}^{n_s - i - 1} \gamma^j r_{t+i+j}\right) \\ + \gamma^{n_s - i} V_\eta^-(s_{t+n_s}) - V_\eta(s_{t+i}), \quad (8)$$

the A2C loss for the policy parameters $\rho$ at time $t$ is

$$\mathcal{L}_t^A(\rho) = -\frac{1}{n_e n_s} \\ \sum_{\text{envs}}^{n_e} \sum_{i=0}^{n_s - 1} \log \pi_\rho(a_{t+i}|s_{t+i}) A_\eta^{t,i,-}(s_{t+i}, a_{t+i}), \quad (9)$$

and the value function loss to learn $\eta$ can be written as

$$\mathcal{L}_t^V(\eta) = \frac{1}{n_e n_s} \sum_{\text{envs}}^{n_e} \sum_{i=0}^{n_s - 1} A_\eta^{t,i}(s_{t+i}, a_{t+i})^2. \quad (10)$$

Lastly, an entropy loss is added to encourage exploration:

$$\mathcal{L}_t^H(\rho) = -\frac{1}{n_e n_s} \sum_{\text{envs}}^{n_e} \sum_{i=0}^{n_s - 1} H(\pi_\rho(\cdot|s_{t+i})), \quad (11)$$

where $H(\cdot)$ is the entropy of a distribution.

## 3. Deep Variational Reinforcement Learning

Fundamentally, there are two approaches to aggregating the history in the presence of partial observability: remembering features of the past or maintaining beliefs.

In most previous work, including ADRQN (Zhu et al., 2017), the current history $(a_{\leq t}, o_{<t})$ is encoded by an RNN, which leads to the recurrent update equation for the latent state $h_t$:

$$h_t = \text{RNNUpdate}_\phi(h_{t-1}, a_{t-1}, o_t). \quad (12)$$

Since this approach is model-free and does not make use of any generative model of the environment, it is unlikely to

approximate belief update steps, instead relying on memory or simple heuristics.

Inspired by the premise that a good way to solve many POMDPs involves (1) estimating the transition and observation model of the environment, (2) performing inference under this model, and (3) choosing an action based on the inferred belief state, we propose deep variational reinforcement learning (DVRL). It extends the RNN-based approach to explicitly support belief inference. Training everything end-to-end shapes the learned model to be useful for the RL task at hand, and not only for predicting observations.

We first explain our baseline architecture and training method in Section 3.1. For a fair comparison, we modify the original architecture of Zhu et al. (2017) in several ways. We find that our new baseline outperforms their reported results in the majority of cases.

In Sections 3.2 and 3.3, we explain our new latent belief state $\hat{b}_t$ and the recurrent update function

$$\hat{b}_t = \text{BeliefUpdate}_{\theta,\phi}(\hat{b}_{t-1}, a_{t-1}, o_t) \quad (13)$$

which replaces Equation (12). Lastly, in Section 3.4, we describe our modified loss function, which allows learning the model jointly with the policy.

### 3.1. Baseline Architecture

We will compare DVRL to an RNN based encoder as shown in Figure 1. While previous work often used $Q$-learning to train the policy (Hausknecht & Stone, 2015; Zhu et al., 2017; Foerster et al., 2016; Narasimhan et al., 2015), we use $n$-step A2C. This avoids drawing entire trajectories from a replay buffer and allows continuous actions.

Furthermore, since A2C interleaves unrolled trajectories and performs a parameter update only every $n_s$ steps, it makes it feasible to maintain an approximately correct latent state. A small bias is introduced by not recomputing the latent state after each gradient update step.

We also modify the implementation of backpropagation-throught-time (BPTT) for $n$-step A2C in the case of policies with latent states. Instead of backpropagating gradients only through the computation graph of the current update involving $n_s$ steps, we set the size of the computation graph independently to involve $n_g$ steps. This leads to an average BPTT-length of $(n_g - 1)/2$.[2] This decouples the bias-variance tradeoff of choosing $n_s$ from the bias-runtime tradeoff of choosing $n_g$. Our experiments show that choosing $n_g > n_s$ greatly improves the agent's performance.

---

[2]This is implemented in PyTorch using the `retain_graph=True` flag in the `backward()` function.

### 3.2. Extending the Latent State

For DVRL, we extend the latent state to be a set of $K$ particles, capturing the uncertainty in the belief state (Thrun, 2000; Silver & Veness, 2010). Each particle consists of the triplet $(h_t^k, z_t^k, w_t^k)$ (Chung et al., 2015). The value $h_t^k$ of particle $k$ is the latent state of an RNN; $z_t^k$ is an additional stochastic latent state that allows us to learn stochastic transition models; and $w_t^k$ assigns each particle an importance weight.

Our belief state $\hat{b}_t$ is thus an approximation of the posterior distribution in our learned model

$$p_\theta(h_{\leq T}, z_{\leq T}, o_{\leq T}|a_{<T}) = p_\theta(h_0) \prod_{t=1}^{T} \Big( p_\theta(z_t|h_{t-1}, a_{t-1})$$

$$p_\theta(o_t|h_{t-1}, z_t, a_{t-1}) \delta_{\psi_\theta^{\text{RNN}}(h_{t-1}, z_t, a_{t-1}, o_t)}(h_t) \Big), \quad (14)$$

with stochastic transition model $p_\theta(z_t|h_{t-1}, a_{t-1})$, decoder $p_\theta(o_t|h_{t-1}, z_t, a_{t-1})$, and deterministic transition function $h_t = \psi_\theta^{\text{RNN}}(h_{t-1}, z_t, a_{t-1}, o_t)$ which is denoted using the Dirac delta distribution $\delta$ and for which we use an RNN. The model is trained to jointly optimise the ELBO and the expected return.

### 3.3. Recurrent Latent State Update

To update the latent state, we proceed as follows:

$$u_{t-1}^k \sim \text{Discrete}\left( \frac{w_{t-1}^k}{\sum_{j=1}^{K} w_{t-1}^j} \right) \quad (15)$$

$$z_t^k \sim q_\phi(z_t^k | h_{t-1}^{u_{t-1}^k}, a_{t-1}, o_t) \quad (16)$$

$$h_t^k = \psi_\theta^{\text{RNN}}(h_{t-1}^{u_{t-1}^k}, z_t^k, a_{t-1}, o_t) \quad (17)$$

$$w_t^k = \frac{p_\theta(z_t^k | h_{t-1}^{u_{t-1}^k}, a_{t-1}) p_\theta(o_t | h_{t-1}^{u_{t-1}^k}, z_t^k, a_{t-1})}{q_\phi(z_t^k | h_{t-1}^{u_{t-1}^k}, a_{t-1}, o_t)}. \quad (18)$$

First, we resample particles based on their weight by drawing ancestor indices $u_{t-1}^k$. This improves model learning (Le et al., 2018; Maddison et al., 2017) and allows us to train the model jointly with the $n$-step loss (see Section 3.4).

For $k = 1, \ldots, K$, new values for $z_t^k$ are sampled from the encoder $q_\phi(z_t^k | h_{t-1}^{u_{t-1}^k}, a_{t-1}, o_t)$ which conditions on the resampled ancestor values $h_{t-1}^{u_{t-1}^k}$ as well as the last actions $a_{t-1}$ and current observation $o_t$. Latent variables $z_t$ are sampled using the reparameterisation trick. The values $z_t^k$, together with $h_{t-1}^{u_{t-1}^k}, a_{t-1}$ and $o_t$, are then passed to the transition function $\psi_\theta^{\text{RNN}}$ to compute $h_t^k$.

The weights $w_t^k$ measure how likely each new latent state value $(z_t^k, h_t^k)$ is under the model and how well it explains the current observation.

**Figure 2:** Overview of DVRL. We do the following $K$ times to compute our new belief $\hat{b}_t$: Sample an ancestor index $u_{t-1}^k$ based on the previous weights $w_{t-1}^{1:K}$ (Eq. 15). Pick the ancestor particle value $h_{t-1}^{u_{t-1}^k}$ and use it to sample a new stochastic latent state $z_t^k$ from the encoder $q_\phi$ (Eq. 16). Compute $h_t^k$ (Eq. 17) and $w_t^k$ (Eq. 18). Aggregate all $K$ values into the new belief $\hat{b}_t$ and summarise them into a vector representation $\hat{h}_t$ using a second RNN. Actor and critic can now condition on $\hat{h}_t$ and $\hat{b}_t$ is used as input for the next timestep. Red arrows denote random sampling, green arrows the aggregation of $K$ values. Black solid arrows denote the passing of a value as argument to a function and black dashed ones the evaluation of a value under a distribution. Boxes indicate neural networks. Distributions are normal or Bernoulli distributions whose parameters are outputs of the neural network.

To condition the policy $\pi_\rho$ and value function $V_\eta$ on the belief $\hat{b}_t = (z_t^k, h_t^k, w_t^k)_{k=1}^K$, we need to summarise the set of particles into a single vector representation $\hat{h}_t$. One option would be to copmute a weighted average over $K$ policies and value functions that each take in a single particle value $(z_t^k, h_t^k)$ – this however would ignore the uncertainty in the latent state after the next action (Littman et al., 1995). Instead, we use a (second) RNN that sequentially takes in each tuple $(z_t^k, h_t^k, w_t^k)$ and outputs $\hat{h}_t$ as its last latent state.

Additional encoders are used for $a_t$, $o_t$ and $z_t$; see Appendix A for details. Figure 2 summarises the entire update step.

### 3.4. Loss Function

To encourage learning a model, we include the term

$$\mathcal{L}_t^{\text{ELBO}}(\theta, \phi) = -\frac{1}{n_e n_s} \sum_{\text{envs}}^{n_e} \sum_{i=0}^{n_s - 1} \log\left(\frac{1}{K} \sum_{k=1}^{K} w_{t+i}^k\right)$$

$$(19)$$

in each gradient update every $n_s$ steps. This leads to the overall loss:

$$\mathcal{L}_t^{\text{DVRL}}(\rho, \eta, \theta, \phi) = \mathcal{L}_t^A(\rho, \theta, \phi) + \lambda^H \mathcal{L}_t^H(\rho, \theta, \phi) +$$
$$\lambda^V \mathcal{L}_t^V(\eta, \theta, \phi) + \lambda^E \mathcal{L}_t^{\text{ELBO}}(\theta, \phi) . \quad (20)$$

Compared to (9), (10) and (11), the losses now also depend on the encoder parameters $\phi$ and model parameters $\theta$, since the policy and value function now condition on the latent states instead of $s_t$. By introducing the $n$-step approximation $\mathcal{L}_t^{\text{ELBO}}$, we can learn $\theta$ and $\phi$ to jointly optimise $\mathcal{L}_t^{\text{ELBO}}$ and the RL loss $\mathcal{L}_t^A + \lambda^H \mathcal{L}_t^H + \lambda^V \mathcal{L}_t^V$.

If we assume that observations and actions are drawn from the stationary state distribution induced by the policy $\pi_\rho$, then $-\mathcal{L}_t^{\text{ELBO}}$ is a stochastic approximation to the action-conditioned ELBO:

$$\frac{1}{T}\mathbb{E}_{p(\tau)} \text{ELBO}_{\text{SMC}}(o_{\leq T}|a_{<T}) =$$

$$\frac{1}{T}\mathbb{E}_{p(\tau)}\mathbb{E}\left[\sum_{t=1}^{T} \log\left(\frac{1}{K}\sum_{k=1}^{K} w_t^k\right)\Bigg| a_{\leq T}\right] , \quad (21)$$

which is a conditional extension of Equation (6), similar to the extension of VAEs by Sohn et al. (2015). To make Equation (21) tractable, we approximate the expectation over $p(\tau)$ by using sampled trajectories from $n_e$ environments. Furthermore, because we assume a stationary state distribution, we can take the sum $\sum_{t=1}^T$ outside of both expectations. This allows us to perform a stochastic gradient update that is based on only $n_s$ summands instead of all $T$, leading to Equation (19) which includes an additional minus sign to account for its minimisation.

The importance of the resampling step (15) in allowing this approximation becomes clear if we compare (21) with the alternative ELBO for the importance weighted autoencoder (IWAE) (Doucet & Johansen, 2009; Burda et al., 2016) that

does not include resampling:

$$\text{ELBO}_{\text{IWAE}}(o_{\leq T}|a_{<T}) = \mathbb{E}\left[\log\left(\frac{1}{K}\sum_{k=1}^{K}\prod_{t=1}^{T}w_t^k\right)\Bigg| a_{\leq T}\right]. \tag{22}$$

Here, the product over time and summation over particles are swapped. Because this loss is not additive over time anymore, we cannot approximate it with shorter parts of the trajectory. This prevents joint optimisation with the RL loss.

## 4. Related Work

Most existing POMDP literature focusses on *planning* algorithms, where the transition and observation functions, as well as a representation of the latent state space, are known (Barto et al., 1995; McAllester & Singh, 1999; Pineau et al., 2003; Ross et al., 2008; Oliehoek et al., 2008; Roijers et al., 2015). In most realistic domains however, these are not known a priori.

There are several approaches that utilise RNNs in POMDPs (Bakker, 2002; Wierstra et al., 2007; Zhang et al., 2015; Heess et al., 2015), including multi-agent settings (Foerster et al., 2016), learning text-based fantasy games (Narasimhan et al., 2015) or, most recently, applied to Atari (Hausknecht & Stone, 2015; Zhu et al., 2017). As discussed in Section 3, our algorithm extends those approaches by enabling the policy to explicitly reason about a model and the belief state.

Another more specialised approach called QMDP-Net (Karkus et al., 2017) learns a value iteration network (VIN) (Tamar et al., 2016) end-to-end and uses it as a transition model for planning. However, a VIN makes strong assumptions about the transition function and in QMDP-Net the belief update must be performed analytically.

The idea to learn a particle filter based policy that is trained using policy gradients was previously proposed by Coquelin et al. (2009). However, they assume a known model and rely on finite differences for gradient estimation. Instead of optimising an ELBO to learn a maximum-likelihood approximation for the latent representation and corresponding transition and observation model, previous work also tried to learn those dynamics using spectral methods (Azizzadenesheli et al., 2016), a Bayesian approach (Ross et al., 2011; Katt et al., 2017), or nonparametrically (Doshi-Velez et al., 2015). However, these approaches do not scale to large or continuous state and observation spaces. For continuous states, actions, and observations with Gaussian noise, a Gaussian process model can be learned (Deisenroth & Peters, 2012). An alternative to learning an (approximate) transition and observation model is to learn a model over trajectories (Willems et al., 1995). However, this is again only possible for small, discrete observation spaces.

Due to the complexity of the learning in POMDPs, previous work already found benefits to using auxiliary losses. Unlike the losses proposed by Lample & Chaplot (2017), we do not require additional information from the environment. The *UNREAL* agent (Jaderberg et al., 2016) is, similarly to our work, motivated by the idea to improve the latent representation by utilising all the information already obtained from the environment. While their work focuses on finding unsupervised auxiliary losses that provide good training signals, our goal is to use the auxiliary loss to better align the network computations with the task at hand by incorporating prior knowledge as an inductive bias.

There is some evidence from recent experiments on the dopamine system in mice (Babayan et al., 2018) showing that their response to ambiguous information is consistent with a theory operating on belief states.

## 5. Experiments

We evaluate DVRL on Mountain Hike and on flickering Atari. We show that DVRL deals better with noisy or partially occluded observations and that this scales to high dimensional and continuous observation spaces like images and complex tasks. We also perform a series of ablation studies, showing the importance of using many particles, including the ELBO training objective in the loss function, and jointly optimising the ELBO and RL losses.

More details about the environments and model architectures can be found in Appendix A together with additional results and visualisations. All plots and reported results are smoothed over time and parallel executed environments. We average over five random seeds, with shaded areas indicating the standard deviation. All RNNs are gated recurrent units (GRUs) (Cho et al., 2014).

### 5.1. Mountain Hike

In this task, the agent has to navigate along a mountain ridge, but only receives noisy measurements of its current location. Specifically, we have $\mathcal{S} = \mathcal{O} = \mathcal{A} = \mathbb{R}^2$ where $s_t = [x_t, y_t]^T \in \mathcal{S}$ and $o_t = [\hat{x}_t, \hat{y}_t]^T \in \mathcal{O}$ are true and observed coordinates respectively and $a_t = [\Delta x_t, \Delta y_t]^T \in \mathcal{A}$ is the desired step. Transitions are given by $s_{t+1} = s_t + \tilde{a}_t + \epsilon_{s,t}$ with $\epsilon_{s,t} \sim \mathcal{N}(\cdot|0, 0.25 \cdot I)$ and $\tilde{a}_t$ is the vector $a_t$ with length capped to $\|\tilde{a}_t\| \leq 0.5$. Observations are noisy with $o_t = s_t + \epsilon_{o,t}$ with $\epsilon_{o,t} \sim \mathcal{N}(\cdot|0, \sigma_o \cdot I)$ and $\sigma_o \in \{0, 1.5, 3\}$. The reward at each timestep is $R_t = r(x_t, y_t) - 0.01\|a_t\|$ where $r(x_t, y_t)$ is shown in Figure 3. The starting position is sampled from $s_0 \sim \mathcal{N}(\cdot|[-8.5, -8.5]^T, I)$ and each episode ends after 75 steps.

DVRL used 30 particles and we set $n_g = 25$ for both RNN and DVRL. The latent state $h$ for the RNN-encoder architecture was of dimension 256 and 128 for both $z$ and $h$ for
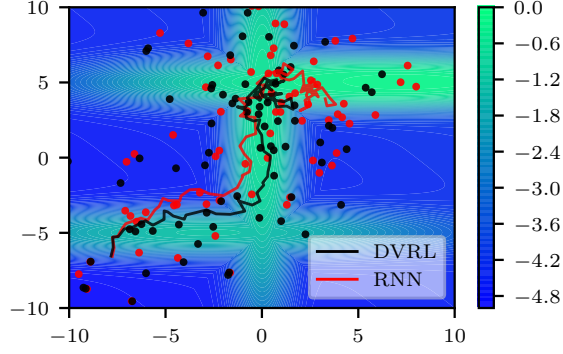
**Figure 3:** Mountain Hike is a continuous control task with observation noise $\sigma_o = 3$. Background colour indicates rewards. Red line: trajectory for RNN based encoder. Black line: trajectory for DVRL encoder. Dots: received observations. Both runs share the same noise values $\epsilon_{i,t}$. DVRL achieves higher returns (see Fig. 4) by better estimating its current location and remaining on the high reward mountain ridge.
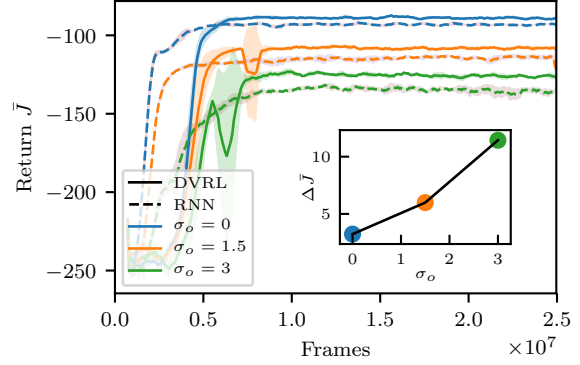


**Figure 4:** Returns achieved in Mountain Hike. Solid lines: DVRL. Dashed lines: RNN. Colour: Noise levels. *Inset:* Difference in performance between RNN and DVRL for same level of noise: $\Delta \bar{J}(\sigma_o) = \bar{J}(\text{DVRL}, \sigma_o) - \bar{J}(\text{RNN}, \sigma_o)$. DVRL achieves slightly higher returns for the fully observable case and, crucially, its performance deteriorates more slowly for increasing observation noise, showing the advantage of DVRL's inference computations in encoding the history in the presence of observation noise.

DVRL. Lastly, $\lambda^E = 1$ and $n_s = 5$ were used, together with RMSProp with a learning rate of $10^{-4}$ for both approaches.

The main difficulty in Mountain Hike is to correctly estimate the current position. Consequently, the achieved return reflects the capability of the network to do so. DVRL outperforms RNN based policies, especially for higher levels of observation noise $\sigma_o$ (Figure 4). In Figure 3 we compare the different trajectories for RNN and DVRL encoders for the same noise, i.e. $\epsilon_{s,t}^{\text{RNN}} = \epsilon_{s,t}^{\text{DVRL}}$ and $\epsilon_{o,t}^{\text{RNN}} = \epsilon_{o,t}^{\text{DVRL}}$ for all $t$ and $\sigma_o = 3$. DVRL is better able to follow the mountain ridge, indicating that its inference based history aggregation is superior to a largely memory/heuristics based one.

The example in Figure 3 is representative but selected for clarity: The shown trajectories have $\Delta J(\sigma_o = 3) = 20.7$ compared to an average value of $\Delta \bar{J}(\sigma_o = 3) = 11.43$ (see Figure 4).

## 5.2. Atari

We chose flickering Atari as evaluation benchmark, since it was previously used to evaluate the performance of ADRQN (Zhu et al., 2017) and DRQN (Hausknecht & Stone, 2015). Atari environments (Bellemare et al., 2013) provide a wide set of challenging tasks with high dimensional observation spaces. We test our algorithm on the same subset of games on which DRQN and ADRQN were evaluated.

Partial observability is introduced by *flickering*, i.e., by a probability of $0.5$ of returning a blank screen instead of the actual observation. Furthermore, only one frame is used as the observation. This is in line with previous work (Hausknecht & Stone, 2015). We use a frameskip of four[3]

---
[3]A frameskip of one is used for Asteroids due to known ren-

and for the stochastic Atari environments there is a $0.25$ chance of repeating the current action for a second time at each transition.

DVRL used 15 particles and we set $n_g = 50$ for both agents. The dimension of $h$ was 256 for both architectures, as was the dimension of $z$. Larger latent states decreased the performance for the RNN encoder. Lastly, $\lambda^E = 0.1$ and $n_s = 5$ was used with a learning rate of $10^{-4}$ for RNN and $2 \cdot 10^{-4}$ for DVRL, selected out of a set of 6 different rates based on the results on ChopperCommand.

Table 1 shows the results for the more challenging stochastic, flickering environments. Results for the deterministic environments, including returns reported for DRQN and ADRQN, can be found in Appendix A. DVRL significantly outperforms the RNN-based policy on five out of ten games and narrowly underperforms significantly on only one. This shows that DVRL is viable for high dimensional observation spaces with complex environmental models.

## 5.3. Ablation Studies

Using more than one particle is important to accurately approximate the belief distribution over the latent state $(z, h)$. Consequently, we expect that higher particle numbers provide better information to the policy, leading to higher returns. Figure 5a shows that this is indeed the case. This is an important result for our architecture, as it also implies that the resampling step is necessary, as detailed in Section 3.4. Without resampling, we cannot approximate the ELBO on only $n_s$ observations.
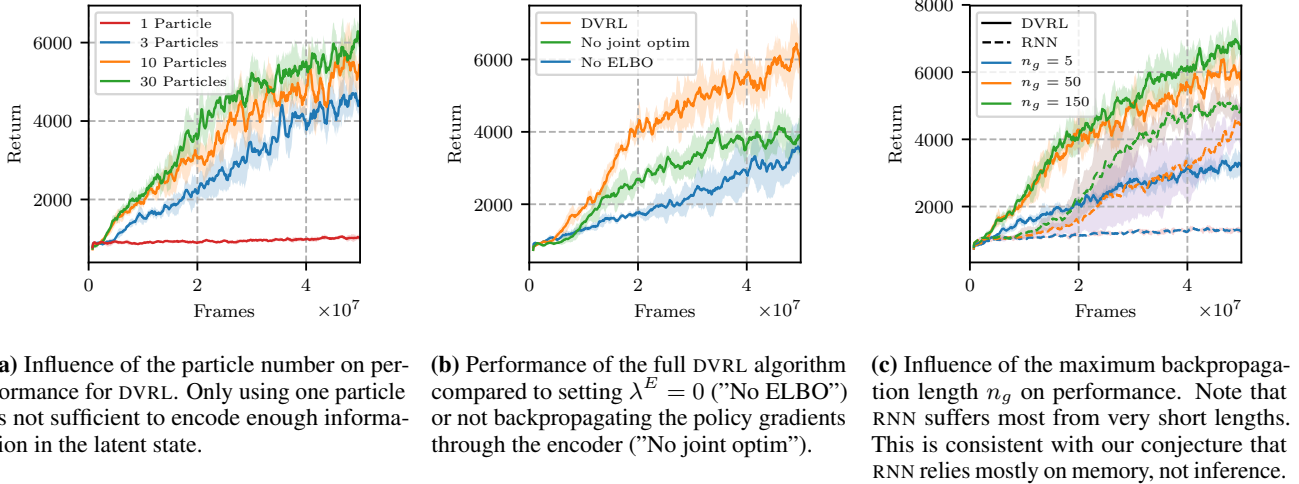
---
dering issues with this environment

**(a)** Influence of the particle number on performance for DVRL. Only using one particle is not sufficient to encode enough information in the latent state.

**(b)** Performance of the full DVRL algorithm compared to setting $\lambda^E = 0$ ("No ELBO") or not backpropagating the policy gradients through the encoder ("No joint optim").

**(c)** Influence of the maximum backpropagation length $n_g$ on performance. Note that RNN suffers most from very short lengths. This is consistent with our conjecture that RNN relies mostly on memory, not inference.

**Figure 5:** Ablation studies on flickering ChopperCommand (Atari).

**Table 1:** Returns on stochastic and flickering Atari environments, averaged over 5 random seeds. Bold numbers indicate statistical significance at the 5% level. Out of ten games, DVRL significantly outperforms the baseline on five games and underperforms narrowly on only one game. Comparisons against DRQN and ADRQN on deterministic Atari environments are in Appendix A.

| Env | DVRL($\pm std$) | RNN($\pm std$) |
| --- | --- | --- |
| Pong | **18.17**($\pm$2.67) | 6.33($\pm$3.03) |
| Chopper | **6602**($\pm$449) | 5150($\pm$488) |
| MsPacman | 2221($\pm$199) | 2312($\pm$358) |
| Centipede | 4240($\pm$116) | 4395($\pm$224) |
| BeamRider | 1663($\pm$183) | 1801($\pm$65) |
| Frostbite | **297**($\pm$7.85) | 254($\pm$0.45) |
| Bowling | 29.53($\pm$0.23) | **30.04**($\pm$0.18) |
| IceHockey | **−4.88**($\pm$0.17) | −7.10($\pm$0.60) |
| DDunk | **−5.95**($\pm$1.25) | −15.88($\pm$0.34) |
| Asteroids | 1539($\pm$73) | 1545($\pm$51) |

Secondly, Figure 5b shows that the inclusion of $\mathcal{L}^{\mathrm{ELBO}}$ to encourage model learning is required for good performance. Furthermore, not backpropagating the policy gradients through the encoder and only learning it based on the ELBO ("No joint optim") also deteriorates performance.

Lastly, we investigate the influence of the backpropagation length $n_g$ on both the RNN and DVRL based policies. While increasing $n_g$ universally helps, the key insight here is that a short length $n_g = 5$ (for an average BPTT-length of 2 timesteps) has a stronger negative impact on RNN than on DVRL. This is consistent with our notion that RNN is mainly performing memory based reasoning, for which longer backpropagation-through-time is required: The belief update (2) in DVRL is a one-step update from $b_t$ to $b_{t+1}$, without the need to condition on past actions and observa-

tions. The proposal distribution can benefit from extended backpropagation lengths, but this is not necessary. Consequently, this result supports our notion that DVRL relies more on inference computations to update the latent state.

## 6. Conclusion

In this paper we proposed DVRL, a method for solving POMDPs given only a stream of observations, without knowledge of the latent state space or the transition and observation functions operating in that space. Our method leverages a new ELBO-based auxiliary loss and incorporates an inductive bias into the structure of the policy network, taking advantage of our prior knowledge that an inference step is required for an optimal solution.

We compared DVRL to an RNN-based architecture and found that we consistently outperform it on a diverse set of tasks, including a number of Atari games modified to have partial observability and stochastic transitions.

We also performed several ablation studies showing the necessity of using an ensemble of particles and of joint optimisation of the ELBO and RL objective. Furthermore, the results support our claim that the latent state in DVRL approximates a belief distribution in a learned model.

Access to a belief distribution opens up several interesting research directions. Investigating the role of better generalisation capabilities and the more powerful latent state representation on the policy performance of DVRL can give rise to further improvements. DVRL is also likely to benefit from more powerful model architectures and a disentangled latent state. Furthermore, uncertainty in the belief state and access to a learned model can be used for curiosity driven exploration in environments with sparse rewards.

## Acknowledgements

## References

Astrom, K. J. Optimal control of markov decision processes with incomplete state estimation. *Journal of mathemati-cal analysis and applications*, 10:174–205, 1965.

Azizzadenesheli, K., Lazaric, A., and Anandkumar, A. Rein-forcement learning of pomdps using spectral methods. In *Proceedings of the 29th Annual Conference on Learning Theory (COLT2016)*, 2016.

Babayan, B. M., Uchida, N., and Gershman, S. J. Belief state representation in the dopamine system. *Nature com-munications*, 9(1):1891, 2018.

Bakker, B. Reinforcement learning with long short-term memory. In *Advances in neural information processing systems*, pp. 1475–1482, 2002.

Barto, A. G., Bradtke, S. J., and Singh, S. P. Learning to act using real-time dynamic programming. *Artificial intelligence*, 72(1-2):81–138, 1995.

Bellemare, M., Veness, J., and Talvitie, E. Skip context tree switching. In *International Conference on Machine Learning*, pp. 1458–1466, 2014.

Bellemare, M. G. Count-based frequency estimation with bounded memory. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation plat-form for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. In *ICLR*, 2016.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. A recurrent latent variable model for sequential data. In *Advances in neural information pro-cessing systems*, 2015.

Coquelin, P.-A., Deguest, R., and Munos, R. Particle filter-based policy gradient in pomdps. In *NIPS*, 2009.

Deisenroth, M. P. and Peters, J. Solving nonlinear con-tinuous state-action-observation pomdps for mechanical systems with gaussian noise. 2012.

Doshi-Velez, F., Pfau, D., Wood, F., and Roy, N. Bayesian nonparametric methods for partially-observable reinforce-ment learning. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):394–407, 2015.

Doucet, A. and Johansen, A. M. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

Foerster, J. N., Assael, Y. M., de Freitas, N., and White-son, S. Learning to communicate to solve riddles with deep distributed recurrent q-networks. *arXiv preprint 1602.02672*, 2016.

Hausknecht, M. and Stone, P. Deep recurrent q-learning for partially observable MDPs. In *2015 AAAI Fall Sympo-sium Series*, 2015.

Heess, N., Hunt, J. J., Lillicrap, T. P., and Silver, D. Memory-based control with recurrent neural networks. *arXiv preprint 1512.04455*, 2015.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforce-ment learning with unsupervised auxiliary tasks. *arXiv preprint 1611.05397*, 2016.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Plan-ning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1), 1998.

Karkus, P., Hsu, D., and Lee, W. S. Qmdp-net: Deep learning for planning under partial observability. In *Ad-vances in Neural Information Processing Systems*, pp. 4697–4707, 2017.

Katt, S., Oliehoek, F. A., and Amato, C. Learning in pomdps with monte carlo tree search. In *International Conference on Machine Learning*, 2017.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *ICLR*, 2014.

Lample, G. and Chaplot, D. S. Playing fps games with deep reinforcement learning. In *AAAI*, pp. 2140–2146, 2017.

Le, T. A., Igl, M., Jin, T., Rainforth, T., and Wood, F. Auto-encoding sequential Monte Carlo. In *ICLR*, 2018.

Littman, M. L., Cassandra, A. R., and Kaelbling, L. P. Learning policies for partially observable environments: Scaling up. In *Machine Learning Proceedings 1995*, pp. 362–370. Elsevier, 1995.

Maddison, C. J., Lawson, J., Tucker, G., Heess, N., Norouzi, M., Mnih, A., Doucet, A., and Teh, Y. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, 2017.

McAllester, D. A. and Singh, S. Approximate planning for factored pomdps using belief state simplification. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, 1999.

McCallum, A. K. and Ballard, D. *Reinforcement learning with selective perception and hidden state*. PhD thesis, University of Rochester. Dept. of Computer Science, 1996.

McCallum, R. A. Overcoming incomplete perception with utile distinction memory. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 190–196, 1993.

Messias, J. V. and Whiteson, S. Dynamic-depth context tree weighting. In *Advances in Neural Information Processing Systems*, pp. 3330–3339, 2017.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

Naesseth, C. A., Linderman, S. W., Ranganath, R., and Blei, D. M. Variational sequential monte carlo. In *AISTATS (To Appear)*, 2018.

Narasimhan, K., Kulkarni, T., and Barzilay, R. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint 1506.08941*, 2015.

Oliehoek, F. A., Spaan, M. T., Whiteson, S., and Vlassis, N. Exploiting locality of interaction in factored dec-pomdps. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*, 2008.

Pineau, J., Gordon, G., Thrun, S., et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, 2003.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Roijers, D. M., Whiteson, S., and Oliehoek, F. A. Point-based planning for multi-objective pomdps. In *IJCAI*, pp. 1666–1672, 2015.

Ross, S., Pineau, J., Paquet, S., and Chaib-Draa, B. Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research*, 32:663–704, 2008.

Ross, S., Pineau, J., Chaib-draa, B., and Kreitmann, P. A bayesian approach for learning and planning in partially observable markov decision processes. *Journal of Machine Learning Research*, 2011.

Shani, G., Brafman, R. I., and Shimony, S. E. Model-based online learning of pomdps. In *European Conference on Machine Learning*, pp. 353–364. Springer, 2005.

Silver, D. and Veness, J. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pp. 2164–2172, 2010.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pp. 3483–3491, 2015.

Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. Value iteration networks. In *Advances in Neural Information Processing Systems*, pp. 2154–2162, 2016.

Thrun, S. Monte carlo pomdps. In *Advances in neural information processing systems*, pp. 1064–1070, 2000.

Wierstra, D., Foerster, A., Peters, J., and Schmidhuber, J. Solving deep memory pomdps with recurrent policy gradients. In *International Conference on Artificial Neural Networks*, pp. 697–706. Springer, 2007.

Willems, F. M., Shtarkov, Y. M., and Tjalkens, T. J. The context-tree weighting method: basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., and Ba, J. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. In *Advances in neural information processing systems*, pp. 5285–5294, 2017.

Zhang, M., Levine, S., McCarthy, Z., Finn, C., and Abbeel, P. Policy learning with continuous memory states for partially observed robotic control. *CoRR*, 2015.

Zhu, P., Li, X., and Poupart, P. On improving deep reinforcement learning for POMDPs. *arXiv preprint 1704.07978*, 2017.